APPENDIX A

```
namespace System.St rage
{
  abstract class It mC ntext : IDisp sabl , IServicePr vid r
  {
```

ItemContext Creation and Management Members

```
// Applications cannot create ItemContext objects directly nor can they derive
// classes from ItemContext.
interal ItemContext();


// Create ItemContext that can be used to search the specified paths or, if no path
// is specified, the default store on the local computer.
public static ItemContext Open();
public static ItemContext Open( string path );
public static ItemContext Open( params string[] paths );


// Return the paths specified when the ItemContext was created.
public string[] GetOpenPaths();


// Create a copy of this ItemContext. The copy will have independent transaction, caching
// and update state. The cache will initially be empty. It is expected that using a
// cloned ItemContext would be more efficient then opening a new ItemContext using the
// same item domain(s).
public ItemContext Clone();


// Close the ItemContext. Any attempt to use the ItemContext after it is closed will
// result in an ObjectDisposedException.
public void Close();
void IDisposable.Dispose();


// True if any domain specified when the ItemConext was opened resolved to a remote
// computer.
public bool IsRemote { get; }


// Returns an object that can provide the requested service type. Returns null if the
// requested service cannot be provided. The use of the IServiceProvider pattern allows
// API that are not normally used and could confuse developers to be factored out of
// the ItemContext class. ItemContext can provide the following kinds of services:
// IItemSerialization, IStoreObjectCache
public object GetService( Type serviceType );
```

Update Related Members

```
// Saves changes represented by all modified objects and all objects passed to
// MarkForCreate or MarkForDelete. May throw UpdateCollisionException if an update
// collision is detected.
public void Update();


// Saves changes represented by the specified objects. The objects must have either
// been modified or passed to MarkForCreate or MarkForDelete, otherwise Argument-
// Exception is thrown. May throw UpdateCollisionException if an update collision is
```

```
// detected.
public void Update( obj  ct obj  ctT  Update );
public void Update( IEnumerabl  obj  ctsToUpdat  );
```

```
// Refreshes the content of the specified objects from the store. If the object has
// been modified, the changes are overwritten and the object is no longer considered
// modified. Throws ArgumentException if anything other then an item, item extension,
// or relationship object is specified.
public void Refresh( object objectToRefresh );
public void Refresh( IEnumerable objectsToRefresh );
```

```
// Raised when an update detects that data has been changed in the store between when a
// modified object was retrieved and an attempt was made to save it. If no event handler
// is registered, the update throws an exception. If an event handler is registered, it
// can throw an exception to abort the update, case the modified object to overwrite
// the data in the store or merge the changes made in the store and in the object.
public event ChangeCollisionEventHandler UpdateCollision;
```

```
// Raised at various points during update processing to provide update progress
// information.
public event UpdateProgressEventhandler UpdateProgress;
```

```
// Async versions of Update
public IAsyncResult BeginUpdate( IAsyncCallback callback, object state );
public IAsyncResult BeginUpdate( object objectToUpdate,
                IAsyncCallback callback,
                object state );
public IAsyncResult BeginUpdate( IEnumerable objectsToUpdate,
                IAsyncCallback callback,
                object state );
public void EndUpdate( IAsyncResult result );
```

```
// Async versions of Refresh
public IAsyncResult BeginRefresh( object objectToRefresh,
                IAsyncCallback callback,
                object state );
public IAsyncResult BeginRefresh( IEnumerable objectsToRefresh,
                IAsyncCallback callback,
                object state );
public void EndRefresh( IAsyncResult result );
```

Transaction Related Members

```
// Begins a transaction with the specified isolation level. The default isolation level
// is ReadCommited. In all cases, a distributed transaction is started because it may
// have to encompass changes stream typed item properties.
public Transaction BeginTransaction();
public Transaction BeginTransaction( System.Data.IsolationLevel isolationLevel );
```

Search Related Members

```
// Create an ItemSearcher that will search this item context for objects of the
// specified type. Throws ArgumentException if a type othern then an item,
// relationship, or item extension is specified.
```

public It mS archer G tS archer(Typ   typ   );

// Find an item given its id.
public Item Findit mByld(It mid it mid);

// Find an item given its path. The path may be absolute or relative. If it is relative,
// NotSupportedException will be thrown if multiple item domains were specified when
// the ItemContext was opened. Will return null if no such item exists. Creates a
// connection to the \\machine\share part of the domain to retrieve the item. The
// item will be assocaited with that domain.
public Item FindItemByPath(string path);

// Find an item given its path. The path is relative to the specified item domain.
// Creates a connection to the specified domain to retrieve the item. The item will be
// associated with that domain. Will return null if no such item exists.
public Item FindItemByPath(string domain, string path);

// Find a set of items given a path. The path is relative to the item domains specified
// when the ItemContext was opened. Will return an empty result if no such item exists.
public FindResult FindAllItemsByPath(string path);

// Find a relationship given its ids.
**public Relatioinship FindRelationshipByld(Itemld itemID,
                                Relationshipld relationshipld);**

// Find a item extension given its ids.
**public ItemExtension FindItemExtensionByld(Itemld itemld,
                                ItemExtensionld itemExtensionld);**

// Find all item, relationship, or item extensions of the specified type optionally
// satisifing a given filter. Throws ArgumentException if a type other then one of
// these is specified.
public FindResult FindAll(Type type);
public FindResult FindAll(Type type, string filter);

// Find any item, relationship, or item extensions of the specified type that satisfies
// a given filter. Throws ArgumentException if a type other then one of these is
// specified. Returns null if no such object is found.
public object FindOne(Type type, string filter);

// Find the item, relationship, or item extensions of the specified type that satisfies
// a given filter. Throws ArgumentException if a type other then one of these is
// specified. Throws ObjectNotFoundException if no such object was found. Throws
// MultipleObjectsFoundException if more then one object was found.
public object FindOnly(Type type, string filter);

// Returns true if an item, relationship, or item extensions of the specified type that
// satisfies a given filter exists. Throws ArgumentException if a type other then one
// of these is specified.
public bool Exists(Type type, string filter);

// Specifies how the objects returned by a search relate to the object identity map
// maintained by the ItemContext.

```
public S archCollisi nMod  SearchCollisi nM  d  { g  t; s  t; }

// Raised when PreserveModifiedObjects is specified for ResultMapping. This event allows
// the application to selectivly update the modified object with data retrieved with the
// search.
public event ChangeCollisionEventHandler SearchCollision;


// Incorporate an object from annother ItemContext into this item context. If an object
// representing the same item, relationship or item extension does not already exist
// this this ItemContext's identity map, a clone of the object is created and added to
// the map. If an object does exist, it is updated with the state and content of the
// specified object in a way concistant with the SearchCollisionMode.
public Item IncorporateItem( Item item );
public Relationship IncorporateRelationship( Relationship relationship );
public ItemExtension IncorporateItemExtension( ItemExtension itemExtension );

}


// Handler for ItemContext.UpdateCollision and ItemSearcher.SearchCollision events.
public delegate void ChangeCollisionEventHandler( object source,
                         ChangeCollisionEventArgs args );


// Arguments for the ChangeCollisionEventHandler delegate.
public class ChangeCollisionEventArgs : EventArgs
{
  // Modified item, item extension, or relationship object.
  public object ModifiedObject { get; }


  // Properties from store.
  public IDictionary StoredProperties { get; }
}


// Handler for ItemContext.UpdateProgress.
public delegate void UpdateProgressEventHandler( ItemContext itemContext,
                         UpdateProgressEventArgs args );


// Arguments for the UpdateProgressEventHandler delegate
public class ChangeCollisionEventArgs : EventArgs
{
  // The current update operation.
  public UpdateOperation CurrentOperation { get; }


  // The object that is currently being updated.
  public object CurrentObject { get; }
}


// Specifies how the objects returned by a search relate to the objects identity map
// maintained by the ItemContext.
public enum SearchCollisionMode
{
  // Indicates that new objects should be created and returned. Objects representing the
  // same item, item extension, or relationship in the identity map are ignored. If this
  // option is specified the SearchCollision event will not be raised.
  DoNotMapSearchResults,
```

- 183 -

```
// Indicates that objects from the identity map should be returned. If the content of
// an object has been modified by the application, the modified object's content is
// preserved. If the object has not been modified, its content is updated with the
// data returned by the search. The Application may provide an handler for the
// SearchCollision event and selectivly update the object as desired.
PreserveModifiedObjects,

// Indicates that the objects from the identity map should be returned. The content
// of the object is updated with the data returned by the search, even if the object
// has been modified by the application. If this option is specified the Search-
// Collision event will not be raised.
OverwriteModifiedObjects
}

// The current update operation.
public enum UpdateOperation
{
// Provided when Update is first called. CurrentObject will be null.
OverallUpdateStarting,

// Provided just before Update returns after a successful update. CurrentObject will be
// null.
OverallUpdateCompletedSucessfully,

// Provided just before Update throws an exception. CurrentObject will be the exception
// object.
OverallUpdateCompletedUnsuccessfully,

// Provided when the update of an object is started. CurrentObject will reference the
// object that will be used for the updated.
ObjectUpdateStaring,

// Provided when a new connection is needed. CurrentObject will be a string that contains
// the path identifying an item domain as passed to ItemContext.Open or retrieved from
// the Location field of a relationship.
OpeningConnection
}
}
```

[Remainder of Page Intentionally Left Blank]